



Embedded SDK (Software Development Kit)

CPT Detection Library

SDK123/D
Rev. 2, 07/17/2002



Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

**For More Information On This Product,
Go to: www.freescale.com**

Contents

About This Document

Audience	ix
Organization	ix
Suggested Reading	ix
Conventions	x
Definitions, Acronyms, and Abbreviations	x
References	xi

Chapter 1

Introduction

1.1 Quick Start	1-1
1.2 Overview of CPT Detection	1-1
1.2.1 Background	1-1
1.2.2 Features and Performance	1-2

Chapter 2

Directory Structure

2.1 Required Core Directories	2-1
2.2 Optional (Domain-Specific) Directories	2-2
2.3 Demo Directory Structure	2-3

Chapter 3

CPT Detection Library Interfaces

3.1 CPT Detection Services	3-1
3.2 Interface	3-1
3.3 Specifications	3-4
3.3.1 CPTDetCreate	3-5
3.3.2 CPTDetInit	3-8
3.3.3 CPTDetection	3-11
3.3.4 CPTDetDestroy	3-12

Chapter 4

Building the CPT Library

4.1 Building the CPT Library	4-1
4.1.1 Dependency Build	4-1
4.1.2 Direct Build	4-2

Chapter 5

Linking Applications with the CPT Library

5.1 CPT Library 5-1
5.1.1 Library Sections 5-1

Chapter 6

CPT Detection Applications

6.1 Test and Demo Applications.....6-1

Chapter 7

License

7.1 Limited Use License Agreement 7-1

List of Tables

Table 1-1	Tone Specifications.	1-2
Table 3-1	CPTDetCreate Arguments	3-5
Table 3-2	CPTDetInit Arguments.	3-8
Table 3-3	CPTDetection Arguments.	3-11
Table 3-4	CPTDetDestroy Arguments	3-12

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Figures

Figure 2-1	Core Directories	2-1
Figure 2-2	DSP56824 Directories	2-2
Figure 2-3	cpt Directory Structure	2-2
Figure 2-4	SDK Application Directory Structure.....	2-3
Figure 2-5	CPT Detection Demo Directory Structure	2-4
Figure 4-1	Dependency Build for CPT Detection Library.....	4-2
Figure 4-2	cpt.mcp Project	4-2
Figure 4-3	Execute Make	4-3

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Examples

Code Example 3-1	C Header File cptdet	3-1
Code Example 3-2	mem Library	3-5
Code Example 3-3	Use of CPTDetCreate Interface	3-6
Code Example 3-4	Sample Callback Procedure	3-9
Code Example 3-5	Use of CPTDetInit Interface	3-9
Code Example 3-6	Use of CPTDetection Interface	3-11
Code Example 3-7	Use of CPTDetDestroy Interface	3-12
Code Example 5-1	linker.cmd File	5-1

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

About This Document

This manual describes the CPT Detection algorithm for use with Motorola's Embedded Software Development Kit, (SDK).

Audience

This document targets software developers implementing CPT Detection within software applications.

Organization

This manual is arranged in the following sections:

- **Chapter 1, Introduction** -- provides a brief overview of this document
- **Chapter 2, Directory Structure** -- provides a description of the required core directories
- **Chapter 3, CPT Detection Library Interfaces** -- describes all of the CPT Detection Library functions
- **Chapter 4, Building the CPT Library** -- tells how to execute the system library project build
- **Chapter 5, Linking Applications with the CPT Library** -- describes the organization of the CPT Detection Library
- **Chapter 6, CPT Detection Applications** -- describes the use of the CPT Detection Library through test/demo applications
- **Chapter 7, License** -- provides the license required to use this product

Suggested Reading

We recommend that you have a copy of the following references:

- *DSP56800 Family Manual*, DSP56800FM/AD
- *DSP56824 User's Manual*, DSP56824UM/AD
- *Inside CodeWarrior: Core Tools*, Metrowerks Corp.

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
Courier Monospaced Type	Commands, command parameters, code examples, expressions, datatypes, and directives	...*Foundational include files... ...a data structure of type <code>vad_tConfigure</code> ...
<i>Italic</i>	Calls, functions, statements, procedures, routines, arguments, file names and applications	...the <i>pConfig</i> argument... ...defined in the C header file, <i>aec.h</i> makes a call to the <i>Callback</i> procedure...
Bold	Reference sources, paths, emphasis	...refer to the Targeting DSP56824 Platform manual.... ... see: C:\Program Files\Motorola\Embedded SDK\help\tutorials
<i>Bold/Italic</i>	Directory name, project name	...and contains these core directories: <i>applications</i> contains applications software.... ...CodeWarrior project, <i>3des.mcp</i> , is.....
Blue Text	Linkable on-line	...refer to Chapter 7 , License...
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES ⁻¹
ALL CAPITAL LETTERS	Variables, directives, defined constants, files libraries	INCLUDE_DSPFUNC #define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]...
Quotation marks "... "	Returned messages	...the message, "Test Passed" is displayed.... ...if unsuccessful for any reason, it will return "NULL"....

Definitions, Acronyms, and Abbreviations

The following list defines the acronyms and abbreviations used in this document. As this template develops, this list will be generated from the document. As we develop more group resources, these acronyms will be easily defined from a common acronym dictionary. Please note that while the acronyms are in solid caps, terms in the definition should be initial capped ONLY IF they are trademarked names or proper nouns.

CPT	Call Progress Tone
DSP	Digital Signal Processor or Digital Signal Processing
I/O	Input/Output
IDE	Integrated Development Environment
LSB	Least Significant Bit

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

MAC	Multiply/Accumulate
MIPS	Million Instructions Per Second
MSB	Most Significant Bit
SDK	Software Development Kit
SRC	Source

References

The following sources were referenced to produce this book:

1. *DSP56800 Family Manual*, DSP56800FM/AD
2. *DSP56824 User's Manual*, DSP56824UM/AD
3. *Embedded SDK Programmer's Guide*

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 1

Introduction

Welcome to Motorola's Family of Digital Signal Processors, (DSPs). This document describes the Call Progress Tone (CPT) Detection Library, which is a part of Motorola's comprehensive Software Development Kit (SDK) for its DSPs. In this document, you will find all the information required to use and maintain the CPT Detection Library interface and algorithms.

Motorola provides these algorithms to you for use on the Motorola DSPs to expedite your application development and reduce the time it takes to bring your own products to market.

Motorola's CPT Detection Library is licensed for your use on Motorola processors. Please refer to the standard Software License Agreement in [Chapter 7](#) for license terms and conditions; please consult with your Motorola representative for premium product licensing.

1.1 Quick Start

Motorola Embedded SDK is targeted to a large variety of hardware platforms. To take full advantage of a particular hardware platform, use **Quick Start** from the **Targeting DSP568xx Platform** documentation.

For example, **Targeting DSP56824 Platform** manual provides more specific information and examples about this hardware architecture. If you are developing an application for a DSP56824EVM board or any other DSP56824 development system, refer to the **Targeting DSP56824 Platform** manual for **Quick Start** or other information specific to the DSP56824.

1.2 Overview of CPT Detection

The detection of Call Progress Tones (CPT) is used in the telephone network. The aim is to detect a signal comprised of two frequencies that represent a specific tone. The present CPT library complies with the North American Telecom standards.

1.2.1 Background

CPT is a set of tones which are used to indicate the current status of a call. A computer system that has a DSP and is equipped for phone-line communications can use the DSP for generating and decoding CPT signals, and for handling typical DSP functions.

Each tone consists of two frequencies and a specific temporal pattern, (ON & OFF periods).

The specifications of each tone are listed in [Table 1-1](#).

Table 1-1. Tone Specifications

Signal Name	Frequency (Hz)	Temporal Pattern	Amplitude
Dial Tone	350+440 (+/- 0.7%)	Steady On	-29 to -10 dbm
Message Waiting Indicator Tone	350+440 (+/- 0.7%)	10 Bursts (0.1SecOn, 0.1 Sec off) then steady On	-29 to -10 dbm
Recall Dial Tone/Confirmation Tone	350+440 (+/- 0.7%)	3 Bursts (0.1 Sec On, 0.1 Sec off) then steady On	-29 to -10 dbm
Line Busy	480+620 (+/- 1.5%)	0.5 Sec On, 0.5 Sec off; repeating	-52 to -21 dbm
Re-order	480+620 (+/- 1.5%)	0.25 Sec On, 0.25 Sec off; repeating	-52 to -21 dbm
Audible Ringing	440+480 (+/- 1.6%)	2 Sec On, 4 Sec off; repeating	-47 to -16 dbm

1.2.2 Features and Performance

The CPT library is multichannel and re-entrant.

For details on Memory and MIPS for a particular DSP, refer to the **Libraries** chapter of the appropriate Targeting manual.

Chapter 2

Directory Structure

2.1 Required Core Directories

Figure 2-1 details required platform directories:

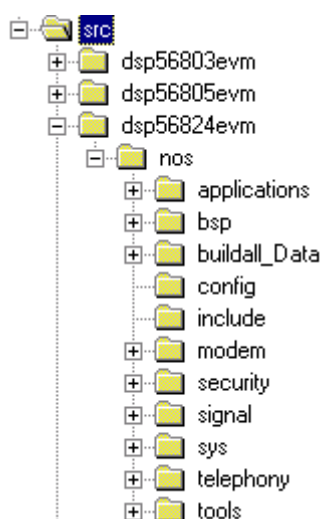


Figure 2-1. Core Directories

As shown in Figure 2-1, DSP56824EVM has no operating system (nos) support and contains these core directories:

- **applications** contains applications software that can be exercised on this platform
- **bsp** contains board support package specific for this platform
- **config** contains default hardware/software configurations for this platform
- **include** contains SDK header files which define the Application Programming Interface
- **sys** contains required system components
- **tools** contains useful utilities used by system components

There are also optional directories that include domain-specific libraries.

2.2 Optional (Domain-Specific) Directories

Figure 2-2 demonstrates how the CPT Detection algorithm is encapsulated in the domain-specific directory *telephony*.

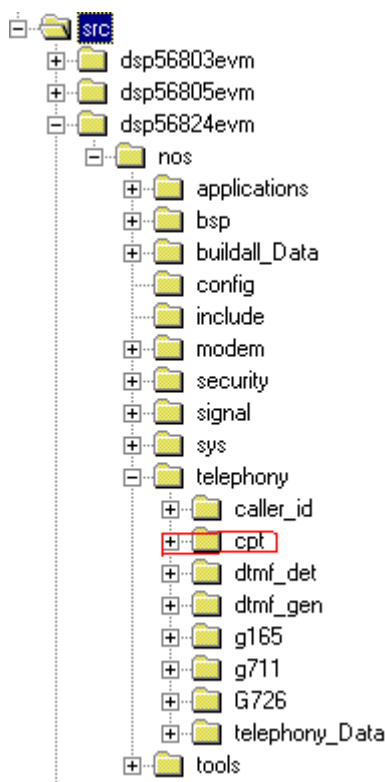


Figure 2-2. DSP56824 Directories

The *cpt* directory includes algorithms specific to CPT Detection. **Figure 2-3** shows the *cpt* directory structure.

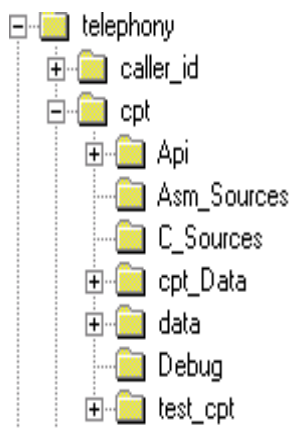


Figure 2-3. cpt Directory Structure

The **cpt** directory includes these sub-directories:

- **Api** includes the APIs for CPT Detection
- **Asm_Sources** includes all asm source files
- **C_Sources** includes all C source files
- **test_cpt** includes C source files and configuration necessary for testing the CPT Detection library modules
 - **C_Sources** contains an example test code for CPT Detection
 - **Config** contains configuration files *appconfig.c*, *appconfig.h* and *linker.cmd* specific to CPT Detection testing.

2.3 Demo Directory Structure

As shown in [Figure 2-4](#), the CPT Detection demo, **demo_cpt**, is encapsulated in the **telephony** directory under **applications**.

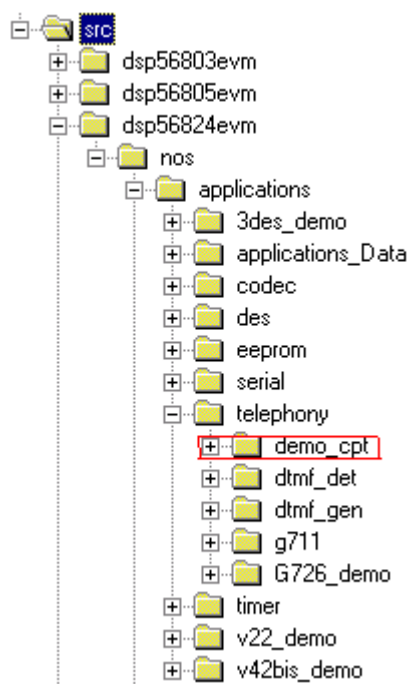


Figure 2-4. SDK Application Directory Structure

The **demo_cpt** directory includes the files specific to the CPT Detection demo. [Figure 2-5](#) shows the directory structure of **demo_cpt**.

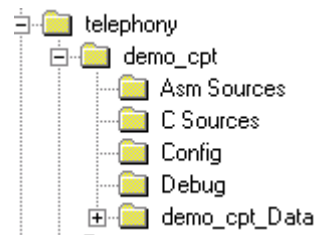


Figure 2-5. CPT Detection Demo Directory Structure

- **demo_cpt** includes the C sources and configuration necessary for testing CPT Detection demo modules
 - **C Sources** contains an example demo code for CPT Detection
 - **Config** contains configuration files **appconfig.c**, **appconfig.h** and **linker.cmd** specific to CPT Detection demo.

Chapter 3

CPT Detection Library Interfaces

3.1 CPT Detection Services

The detection of Call Progress Tones (CPT) is used in the telephone network. The aim is to detect a signal comprised of two frequencies that represent a specific tone.

3.2 Interface

The C interface for the CPT Detection library services is defined in the C header file *cptdet*, shown in [Code Example 3-1](#) as a reference

Code Example 3-1. C Header File *cptdet*

SDK defines the CPT Detection library:
CPT Detection Library (cpt.lib)
The interfaces and services that they provide are described below.

```
/* File cptdet.h */
#ifndef _cptdet_H
#define _cptdet_H

/* Definitions Used */
/* The variable "return_value" used in CPTDet_sCallback
   below uses one of the following definitions
   */

#define DIAL_TONE_DETECTED          0x11
#define MSG_WAITING_TONE_DETECTED  0x12
#define RECALL_DIAL_TONE_DETECTED  0x13
#define BUSY_TONE_DETECTED          0x14
#define REORDER_TONE_DETECTED       0x15
#define RINGING_TONE_DETECTED       0x16

/* #defines to be used by CPT detect algorithm only */
#define NO_FIL                      10
#define ANA_BUF_SIZE                104
#define LAST_BUF_SIZE               24
```

```
#define NO_FIL_CPT          6
#define NC_CPT              80

/*****
    Foundational Include Files
    *****/
#include "port.h"

/*****
    Structure for CPT  Detector Callback
    *****/

typedef struct
{
    void    (*pCallback) ( void    * pCallbackArg,
                          UWord16  return_value);

    void    *pCallbackArg;
} CPTDet_sCallback;

/*****
    Structure for CPT  Detector Configuration
    *****/

typedef struct
{
    CPTDet_sCallback  CPTDetCallback;
}CPTDet_sConfigure;

/*****
    Structure for CPT  Detector Handle
    *****/

typedef struct
{
    CPTDet_sCallback *pCallback;
    Int16    alfa;
    Int16    cpt_on_timer;
    Int16    cpt_off_timer;
    Int16    cpt_state;
    Int16    cpt_last_on;
    Int16    cpt_last_off;
    Int16    cpt_status;
    Int16    cpt_bursts;
    Int16    cpt_last_state;
    Int16    cpt_last_code;
    Int16    cpt_last_group;
    Int16    previous_cpt;

    Int16    sil_on_timer;
    Int16    sil_off_timer;
    Int16    sil_status;
    Int16    cpt_level[2];
}
```

```

Int16    n_e[2];
Int16    decimate_flag;

UWord16  *psik;
Int16    loop_cntr;

Int16    mg_energy[NO_FIL_CPT];

UWord16  *psig_energy;

Int16    ANA_BUF[NC_CPT];
Int16    shift_count;
Int16    ana_buf_ptr;
Int16    ana_buf_count;

Int16    result_ptr;
Int16    frame_info[4];

}CPTDet_sHandle;

/*****
Function Prototypes
*****/

EXPORT CPTDet_sHandle * CPTDetCreate (CPTDet_sConfigure * pConfig);

EXPORT void    CPTDetInit ( CPTDet_sHandle      * pCPTDet,
                           CPTDet_sConfigure    * pConfig);

EXPORT Result   CPTDetection ( CPTDet_sHandle      * pCPTDet,
                               Word16               * pSamples,
                               UWord16              NumberSamples);

EXPORT void    CPTDetDestroy(CPTDet_sHandle * pCPTDet);

#endif

```

3.3 Specifications

The following pages describe the CPT Detection library functions.

Function arguments for each routine are described as *in*, *out*, or *inout*. An *in* argument means that the parameter value is an input only to the function. An *out* argument means that the parameter value is an output only from the function. An *inout* argument means that a parameter value is an input to the function, but the same parameter is also an output from the function.

Typically, *inout* parameters are input pointer variables in which the caller passes the address of a pre-allocated data structure to a function. The function stores its results within that data structure. The actual value of the *inout* pointer parameter is not changed.

3.3.1 CPTDetCreate

Call(s):

```
EXPORT CPTDet_sHandle *CPTDetCreate (CPTDet_sConfigure *pConfig);
```

Required Header: "cptdet.h"

Arguments:

Table 3-1. CPTDetCreate Arguments

<i>pConfig</i>	in	Points to the configuration data for CPT Detection
----------------	----	--

Description: The *CPTDetCreate* function creates an instance of the CPT Detection.

The *pConfig* argument points to *CPTDet_sConfigure*, the structure used to configure CPT Detection operation. For initialization of the *CPTDet_sConfigure* structure, please refer to *CPTDetInit*, [Section 3.3.2](#). A total of 134 words are allocated per instance. This function itself calls the *CPTDetInit* () function, so the user need not call *CPTDetInit*. The library allocates dynamic memory using the mem library, shown in [Code Example 3-2](#). The CPT library is **multichannel** and **re-entrant**.

Code Example 3-2. mem Library

```
#include "arch.h"
#include "mem.h"
#include "cptdet.h"

CPTDet_sHandle * CPTDetCreate (CPTDet_sConfigure *pConfig)
{
    CPTDet_sHandle *pCPTDet;
    bool flag = true;

    pCPTDet = (CPTDet_sHandle *) memMallocEM (sizeof (CPTDet_sHandle));
    if (pCPTDet == NULL)
        return (NULL);

    pCPTDet->psig_energy = (UWord16 *) memMallocAlignedEM ( 2 * sizeof
                                                             (Word16));
    if (pCPTDet->psig_energy == NULL)
        flag = false;

    if (memIsAligned (pCPTDet->psig_energy,2 ) == NULL)
        flag = false;

    pCPTDet->psik = (UWord16 *) memMallocAlignedEM ( 2 * NO_FIL * sizeof (
                                                         Word16));
    if (pCPTDet->psik == NULL)
        flag = false;

    if (memIsAligned (pCPTDet->psik,2 * NO_FIL) == NULL)
        flag = false;

    pCPTDet->pCallback = (CPTDet_sCallback *) memMallocEM (sizeof
                                                            (CPTDet_sCallback));
    if (pCPTDet->pCallback == NULL)
        flag = false;
```

```

    if (flag == false)
    {
        CPTDetDestroy (pCPTDet);
        return (NULL);
    }

    CPTDetInit (pCPTDet, pConfig);

    return (pCPTDet);
}

```

For details on the *CPTDet_sHandle* structure, refer to [Code Example 3-1](#).

If the *CPTDetCreate* function is called to create an instance, then *CPTDetDestroy* (see [Section 3.3.4](#)) should be used to destroy the instance.

Alternatively, the user can allocate memory statically, which requires duplicating all the statements in the *CPTDetCreate* function. In this case, the user can call the *CPTDetInit* function directly, bypassing the *CPTDetCreate* function.

If the user dynamically allocates memory without calling *CPTDetCreate*, then the user himself must destroy the memory allocated.

Returns: A pointer to an instance of CPT Detection.

Special Considerations:

- The CPT application is multichannel and re-entrant.
- If *CPTDetCreate* is called, then the user need not call *CPTDetInit* function as it is called internally in the *CPTDetCreate* function.

Code Example 3-3. Use of *CPTDetCreate* Interface

```

#include "cptdet.h"
#include "mem.h"

/* Function prototypes */
void CPTDetCallback ( void *pCallbackArg, UWord16 return_value);

void test_CPTdet (void)
{
    CPTDet_sHandle *pCPTDet;

    CPTDet_sConfigure *pConfig;

    pConfig = (CPTDet_sConfigure *) memMallocEM (sizeof (CPTDet_sConfigure));

    if (pConfig == NULL) assert(!"Out of memory");

    /* Initialize the callback function */

    pConfig->CPTDetCallback.pCallback = CPTDetCallback;

```

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
pCPTDet = CPTDetCreate (pConfig);

/* Create and initialize the pConfig structure */
....
pCPTDet = CPTDetCreate (pConfig);
....
}

.
```

3.3.2 CPTDetInit

Call(s):

```
Result CPTDetInit (CPTDet_sHandle *pCPTDet, CPTDet_sConfigure *pConfig);
```

Required Header: "cptdet.h"

Arguments:

Table 3-2. CPTDetInit Arguments

<i>pCPTDet</i>	in	Handle to an instance of CPT Detection
<i>pConfig</i>	in	A pointer to a data structure containing data for initializing the CPT Detection algorithm

Description: The *CPTDetInit* function will initialize the CPT Detection algorithm. Before calling the *CPTDetInit* function, a CPT instance must be created, either by calling the *CPTDetCreate* function, (see [Section 3.3.1](#)), or by allocating the memory statically, which does not require calling the *CPTDetCreate* function. [Section 3.3.1](#) also offers details on dynamic and static memory allocation. During initialization, all resources will be set to their initial values in preparation for CPT Detection operation.

The parameter *pConfig* points to a data structure of type *CPTDet_sConfigure*; its fields initialize CPT Detection operation in the following manner:

Callback - A structure of type *CPTDet_sCallback*; it describes the procedure CPT detect will call once a valid CPT is detected.

```
void (*pCallback) (void *pCallbackArg, UWord16 return_value
```

The callback procedure parameter, *pCallbackArg*, is supplied by the user in the *CPTDet_sCallback* structure; this value is passed back to the user during the call to the Callback procedure. Typically, *pCallbackArg* points to context information used by the callback procedure, which the user must write.

return_value indicates one of the the CPT tones detected. the return_value contains one of the following #defines.

```
#define DIAL_TONE_DETECTED          0x11
#define MSG_WAITING_TONE_DETECTED   0x12
#define RECALL_DIAL_TONE_DETECTED   0x13
#define BUSY_TONE_DETECTED          0x14
#define REORDER_TONE_DETECTED       0x15
#define RINGING_TONE_DETECTED       0x16
```

An example callback procedure is shown as a reference in [Code Example 3-4](#) and prints the CPT detected. You must write your own callback procedure.

Code Example 3-4. Sample Callback Procedure

```
#include "CPTdet.h"

void CPTDetCallback (void *pCallbackArg, UWord16 return_value)
{
    switch (return_value)
    {
        case (DIAL_TONE_DETECTED):
            printf("Dial tone detected \n\n");
            break;
        case (MSG_WAITING_TONE_DETECTED):
            printf("Message Waiting tone detected \n\n");
            break;
        case (RECALL_DIAL_TONE_DETECTED):
            printf ("Recall Dial tone detected \n\n");
            break;
        case (BUSY_TONE_DETECTED):
            printf ("Busy tone detected \n\n");
            break;
        case (REORDER_TONE_DETECTED):
            printf ("Reorder tone detected \n\n");
            break;
        case (RINGING_TONE_DETECTED):
            printf ("Audible ringing tone detected \n\n");
            break;
        default:
            printf ("Tone not Detected \n\n");
            break;
    }
}
```

Special Considerations:

- If *CPTDetCreate* is called, then the user need not call *CPTDetInit* function, as it is called internally in the *CPTDetCreate* function.

Code Example 3-5. Use of *CPTDetInit* Interface

```
#include "cptdet.h"
#include "mem.h"

/* Function prototypes */
void CPTDetCallback ( void *pCallbackArg, UWord16 return_value);

void test_CPTdet (void)
{
    CPTDet_sHandle *pCPTDet;

    CPTDet_sConfigure *pConfig;
```

```
pConfig = (CPTDet_sConfigure *) memMallocEM (sizeof (CPTDet_sConfigure));

if (pConfig == NULL) assert(!"Out of memory");

/* Initialize the callback function */

pConfig->CPTDetCallback.pCallback = CPTDetCallback;

/* Create and initialize the pConfig structure */
....
pCPTDet = CPTDetCreate (pConfig); /* This itself calls CPTDetInit function */
....
}
```

3.3.3 CPTDetection

Call(s):

```
Result CPTDetection (CPTDet_sHandle *pCPTDet, Word16 *pSamples, UWord16
                    NumSamples);
```

Required Header: "cptdet.h"**Arguments :****Table 3-3. CPTDetection Arguments**

<i>pCPTDet</i>	in	Handle to an instance of CPT Detection
<i>pSamples</i>	in	Pointer to Input buffer containing 16-bit linear PCM samples which are in 1.15 (Q15) format
<i>NumSamples</i>	in	The number of CPT samples as input

Description: The *CPTDetection* function detects whether the CPT input samples represent a "Valid Tone"; the samples are in 1.15 format.

Returns: Upon successful detection, *CPTDetection* calls, then passes the detected tone to the callback function. If a tone is detected, *CPTDetection* will return "PASS" to the calling function; otherwise, it returns "FAIL".

Special Considerations: The *CPTDetection* function expects linear 16bit-PCM samples, sampled at 8 KHz only. As soon as a valid CPT is detected, this function calls the user callback procedure and exits from further processing of samples.

Code Example 3-6. Use of CPTDetection Interface

```
#include "cptdet.h"
#include "mem.h"

#define BUFSIZE 160
/* Function prototypes */
void CPTDetCallback ( void *pCallbackArg, UWord16 return_value);

void test_CPTdet (void)
{
    Result result;
    UWord16 NumSamples;
    Word16 SampleBuf[160];
    CPTDet_sHandle *pCPTDet;
    CPTDet_sConfigure *pConfig;

    pConfig = (CPTDet_sConfigure *) memMallocEM (sizeof (CPTDet_sConfigure));

    if (pConfig == NULL) assert(!"Out of memory");

    /* Initialize the callback function */
    pConfig->CPTDetCallback.pCallback = CPTDetCallback;

    /* Create and initialize the pConfig structure */
    pCPTDet = CPTDetCreate (pConfig); /* This itself calls CPTDetInit function */
    NumSamples = BUFSIZE;
    result = CPTDetection(pCPTDet, SampleBuf, NumSamples);
}
```

3.3.4 CPTDetDestroy

Call(s):

```
void CPTDetDestroy (CPTDet_sHandle *pCPTDet);
```

Required Header: "cptdet.h"

Arguments:

Table 3-4. CPTDetDestroy Arguments

<i>pCPTDet</i>	in	Handle to an instance of CPT Detect created by a call to <i>CPTDetCreate</i>
----------------	----	--

Description: The *CPTDetDestroy* function frees the instance of *CPTDetect* originally created by a call to *CPTDetCreate*.

Returns: None

Special Considerations: None

Code Example 3-7. Use of CPTDetDestroy Interface

```
#include "cptdet.h"
#include "mem.h"

#define BUFSIZE 160
/* Function prototypes */
void CPTDetCallback ( void *pCallbackArg, UWord16 return_value);

void test_CPTdet (void)
{
    Result result;
    UWord16 NumSamples;
    Word16 SampleBuf[160];
    CPTDet_sHandle *pCPTDet;
    CPTDet_sConfigure *pConfig;

    pConfig = (CPTDet_sConfigure *) memMallocEM (sizeof (CPTDet_sConfigure));

    if (pConfig == NULL) assert(!"Out of memory");

    /* Initialize the callback function */

    pConfig->CPTDetCallback.pCallback = CPTDetCallback;

    /* Create and initialize the pConfig structure */
    pCPTDet = CPTDetCreate (pConfig); /* This itself calls CPTDetInit function */
    NumSamples = BUFSIZE;

    result = CPTDetection (pCPTDet, SampleBuf, NumSamples);

    CPTDetDestroy (pCPTDet);
}
```

Chapter 4

Building the CPT Library

4.1 Building the CPT Library

The CPT library combines all of the components described in previous sections into one library: *cpt.lib*. To build this library, a Metrowerks' CodeWarrior project, *cpt.mcp*, is provided. This project and all the necessary components to build the CPT Generation library are located in the\nos\telephony\cpt directory of the SDK directory structure.

There are two methods to execute a system library project build: dependency build and direct build.

4.1.1 Dependency Build

Dependency build is the easiest approach and requires no additional work on the user's part. If you add the CPT library project, *cpt.mcp*, to your application project as shown in [Figure 4-1](#), the CPT library will automatically build when the application is built.

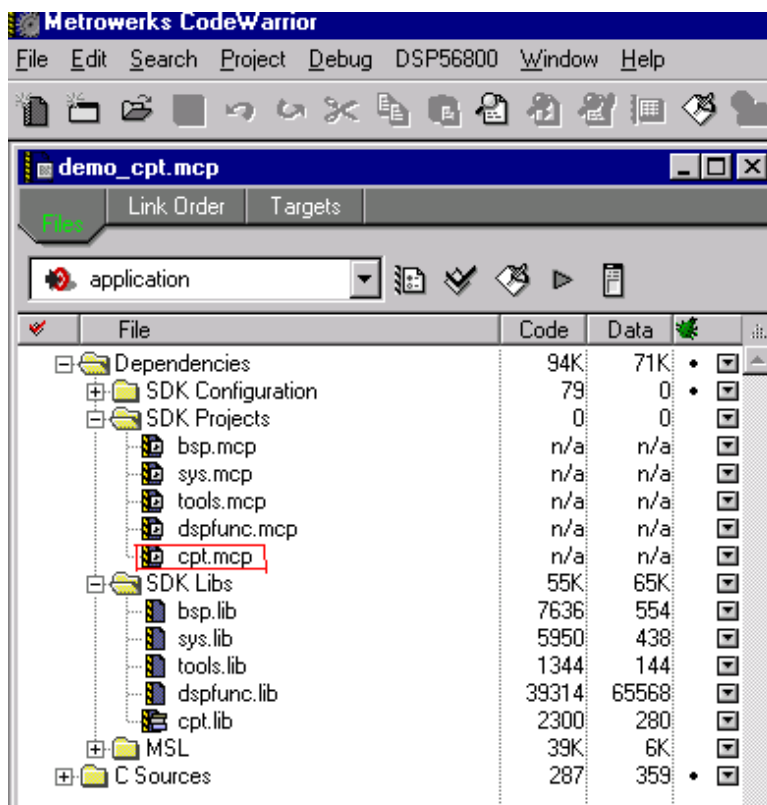


Figure 4-1. Dependency Build for CPT Detection Library

4.1.2 Direct Build

Direct build allows you to build the CPT Detection library independently of any other build. Follow these steps:

Step 1. Open *cpt.mcp* project, as shown in Figure 4-2.

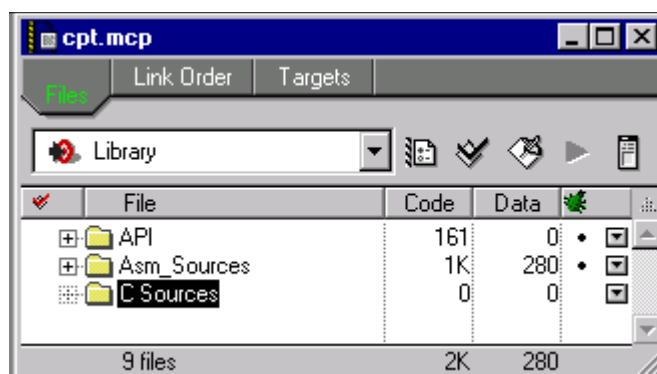


Figure 4-2. *cpt.mcp* Project

Step 2. Execute the build by pressing function key [F7] or by choosing the *Make* command from the Project menu; see Figure 4-3.

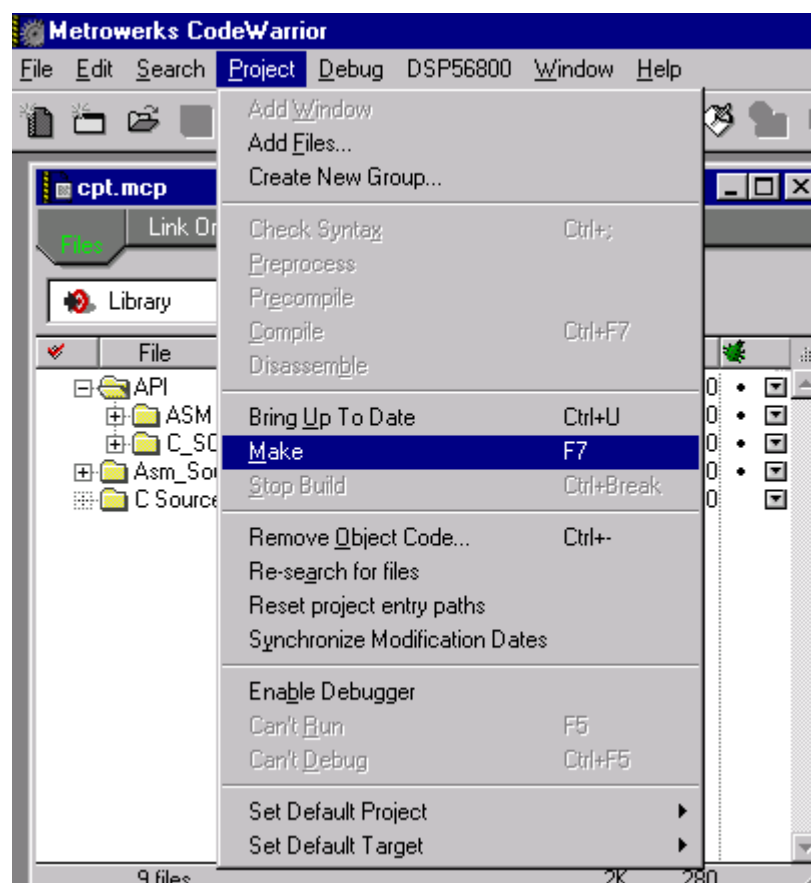


Figure 4-3. Execute Make

At this point, the if build is successful, the *cpt.lib* library file is created in the ...\\nos\\telephony\\cpt\\Debug directory.

Chapter 5

Linking Applications with the CPT Library

5.1 CPT Library

The CPT library includes APIs, which provide interface between the user application and the CPT modules. To invoke CPT Detection, APIs must be called in this order:

- CPTDetCreate (.....); /* should be called to create an instance(s) of CPT*/
- CPTDetInit (.....) /* called within the CPTDetCreate function */
- CPTDetection (.....); /* can be called per the number of samples */
- CPTDetDestroy (.....); /* should be called to destroy the instance(s) of CPT which are created by a CPTDetCreate () function */

5.1.1 Library Sections

The CPT Library contains the following data ROM section that needs to be placed in memory through linker command file.

cpt_data - Data ROM

See <Link>**Code Example 5-1** for a sample *linker.cmd* file, which may be used for testing the DES library.

Code Example 5-1. *linker.cmd* File

```
# Linker.cmd file for DSP56824EVM External RAM
# using both internal and external data memory (EX = 0)
# and using external program memory (Mode = 3)

MEMORY {
    .pram    (RWX) : ORIGIN = 0x0000, LENGTH = 0xFF80 # ? external program memory
    .avail   (RW)  : ORIGIN = 0x0000, LENGTH = 0x0030 # available
    .cwregs  (RW)  : ORIGIN = 0x0030, LENGTH = 0x0010 # C temp registers in
    CodeWarrior
    .im1     (RW)  : ORIGIN = 0x0040, LENGTH = 0x07C0 # data 1
    .rom     (R)   : ORIGIN = 0x0800, LENGTH = 0x0800 # internal data ROM
```

```
.im2    (RW) : ORIGIN = 0x1000, LENGTH = 0x0600 # data 2
.hole   (R)  : ORIGIN = 0x1600, LENGTH = 0x0A00 # hole
.data   (RW) : ORIGIN = 0x2000, LENGTH = 0xC000 # data segment
.em      (RW) : ORIGIN = 0xE000, LENGTH = 0x1000 # data 3
.stack  (RW) : ORIGIN = 0xF000, LENGTH = 0x0F80 # stack
.onchip1(RW) : ORIGIN = 0xFF80, LENGTH = 0x0040 # on-chip peripheral
registers
.onchip2(RW) : ORIGIN = 0xFFC0, LENGTH = 0x0040 # on-chip peripheral
registers
```

```
}
```

```
FORCE_ACTIVE {FconfigInterruptVector}
```

```
SECTIONS {
```

```
#
# Data (X) Memory Layout
#
    _EX_BIT      = 0;

    # Internal Memory Partitions (for mem.h partitions)
    _NUM_IM_PARTITIONS = 2; # .im1 and .im2

    # External Memory Partition (for mem.h partitions)
    _NUM_EM_PARTITIONS = 1; # .em

    .main_application_code :
    {
        # .text sections

        # config.c MUST be placed first, otherwise the Interrupt Vector
        # configInterruptVector will not be located at the correct
        # address, P:0x0000

        config.c (.text)
        * (.text)
        * (rtlib.text)
        * (fp_engine.text)
        * (user.text)
    } > .pram

    .main_application_data :
    {
        #
        # Define variables for C initialization code
        #
        F_Xdata_start_addr_in_ROM = ADDR(.rom) + SIZEOF(.rom) / 2;
        F_StackAddr              = ADDR(.stack);
        F_StackEndAddr           = ADDR(.stack) + SIZEOF(.stack) / 2 - 1;

        F_Xdata_start_addr_in_RAM = .;

        #
        # Memory layout data for SDK INCLUDE_MEMORY (mem.h) support
        #

        FmemEXbit = .;
        WRITEH(_EX_BIT);
        FmemNumIMpartitions = .;
        WRITEH(_NUM_IM_PARTITIONS);
        FmemNumEMpartitions = .;
```

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```

        WRITEH(_NUM_EM_PARTITIONS);
FmemIMpartitionList = .;
#    WRITEH(ADDR(.im1));
#    WRITEH(SIZEOF(.im1) / 2);
        WRITEH(ADDR(.im2));
        WRITEH(SIZEOF(.im2) / 2);
FmemEMpartitionList = .;
        WRITEH(ADDR(.em));
        WRITEH(SIZEOF(.em) / 2);

# .data sections

* (.data)
* (fp_state.data)
* (rtlib.data)

```

CPT external data begins here

#-----

* (cpt_data.data)

CPT external data ends here

#-----

F_Xdata_ROMtoRAM_length = 0;

F_bss_start_addr = .;

_BSS_ADDR = .;

* (rtlib.bss.lo)

* (.bss)

F_bss_length = . - _BSS_ADDR; # Copy DATA

} > .data

FArchIO = ADDR(.onchip2);

}

Chapter 6

CPT Detection Applications

6.1 Test and Demo Applications

To verify the CPT Detection algorithm, test and demo applications have been developed. Refer to the **Targeting Motorola DSP568xx Platform** Manual for the DSP you are using to see if the test and demo applications are available for your target.

Chapter 7

License

7.1 Limited Use License Agreement

LIMITED USE LICENSE AGREEMENT

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THIS SOFTWARE. BY USING OR COPYING THE SOFTWARE, YOU AGREE TO THE TERMS OF THIS AGREEMENT.

The software in either source code form ("Source") or object code form ("Object") (cumulatively hereinafter "Software") is provided under a license agreement ("Agreement") as described herein. Any use of the Software including copying, modifying, or installing the Software so that it is usable by or accessible by a central processing unit constitutes acceptance of the terms of the Agreement by the person or persons making such use or, if employed, the employer thereof ("Licensee") and if employed, the person(s) making such use hereby warrants that they have the authority of their employer to enter this license agreement. If Licensee does not agree with and accept the terms of this Agreement, Licensee must return or destroy any media containing the Software or materials related thereto, and destroy all copies of the Software.

The Software is licensed to Licensee by Motorola Incorporated ("Motorola") for use under the terms of this Agreement. Motorola retains ownership of the Software. Motorola grants only the rights specifically granted in this Agreement and grants no other rights. Title to the Software, all copies thereof and all rights therein, including all rights in any intellectual property including patents, copyrights, and trade secrets applicable thereto, shall remain vested in Motorola.

For the Source, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to use, copy, and make derivatives of the Source solely in a development system environment in order to produce object code solely for operating on a Motorola semiconductor device having a central processing unit ("Derivative Object").

For the Object and Derivative Object, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to copy, use, and distribute the Object and the Derivative Object solely for operating on a Motorola semiconductor device having a central processing unit.

Licensee agrees to: (a) not use, modify, or copy the Software except as expressly provided herein, (b) not distribute, disclose, transfer, sell, assign, rent, lease, or otherwise make available the Software, any derivatives thereof, or this license to a third party except as expressly provided herein, (c) not remove, obliterate, or otherwise defeat any copyright, trademark, patent or proprietary notices, related to the Software (d) not in any form export, re-export, resell, ship or divert or cause to be exported, re-exported, resold, shipped, or diverted, directly or indirectly, the Software or a direct product thereof to any country which the United States government or any agency thereof at the time of export or re-export requires an export license or other government approval without first obtaining such license or approval.

License

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

THE SOFTWARE IS PROVIDED ON AN "AS IS" BASIS AND WITHOUT WARRANTY OF ANY KIND INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MOTOROLA BE LIABLE FOR ANY LIABILITY OR DAMAGES OF ANY KIND INCLUDING, WITHOUT LIMITATION, DIRECT OR INDIRECT OR INCIDENTAL OR CONSEQUENTIAL OR PUNITIVE DAMAGES OR LOST PROFITS OR LOSS OF USE ARISING FROM USE OF THE SOFTWARE OR THE PRODUCT REGARDLESS OF THE FORM OF ACTION OR THEORY OF LIABILITY (INCLUDING WITHOUT LIMITATION, ACTION IN CONTRACT, NEGLIGENCE, OR PRODUCT LIABILITY) EVEN IF MOTOROLA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THIS DISCLAIMER OF WARRANTY EXTENDS TO LICENSEE OR USERS OF PRODUCTS AND IS IN LIEU OF ALL WARRANTIES WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE.

Motorola does not represent or warrant that the Software is free of infringement of any third party patents, copyrights, trade secrets, or other intellectual property rights or that Motorola has the right to grant the licenses contained herein. Motorola does not represent or warrant that the Software is free of defect, or that it meets any particular requirements or need of the Licensee, or that it conforms to any documentation, or that it meets any standards.

Motorola shall not be responsible to maintain the Software, provide upgrades to the Software, or provide any field service of the Software. Motorola reserves the right to make changes to the Software without further notice to Licensee.

The Software is not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Software could create a situation where personal injury or death may occur. Should Licensee purchase or use the Software for any such unintended or unauthorized application, Licensee shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the Software.

The term of this Agreement is for as long as Licensee uses the Software for its intended purpose and is not in default of any provisions of this Agreement. Motorola may terminate this Agreement if Licensee is in default of any of the terms and conditions of this Agreement.

This Agreement shall be governed by and construed in accordance with the laws of the State of Arizona and can only be modified in a writing signed by both parties. Licensee agrees to jurisdiction and venue in the State of Arizona.

By using, modifying, installing, compiling, or copying the Software, Licensee acknowledges that this Agreement has been read and understood and agrees to be bound by its terms and conditions. Licensee agrees that this Agreement is the complete and exclusive statement of the agreement between Licensee and Motorola and supersedes any earlier proposal or prior arrangement, whether oral or written, and any other communications relative to the subject matter of this Agreement.

Index

C

CPT [x](#), [1-1](#)
CPT Detection [1-1](#)
cpt_data [5-1](#)
CPTDetCreate [3-5](#)
CPTDetDestroy [3-12](#)
CPTDetection [3-11](#)
CPTDetInit [3-8](#)

D

DSP [x](#)
DSP56800 Family Manual [xi](#)
DSP56824 User's Manual [xi](#)

E

Embedded SDK Programmer's Guide [xi](#)

I

I/O [x](#)
IDE [x](#)

L

LSB [x](#)

M

MAC [xi](#)
MIPS [xi](#)
MSB [xi](#)

S

SDK [xi](#)
SRC [xi](#)

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

**For More Information On This Product,
Go to: www.freescale.com**

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and the Stylized M Logo are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

MOTOROLA and the Stylized M Logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners. © Motorola, Inc. 2002.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140 or 1-800-441-2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3-20-1, Minami-Azabu. Minato-ku, Tokyo 106-8573 Japan. 81-3-3440-3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852-26668334

Technical Information Center: 1-800-521-6274

HOME PAGE: <http://www.motorola.com/semiconductors/>

**MOTOROLA**

**For More Information On This Product,
Go to: www.freescale.com**

SDK123/D